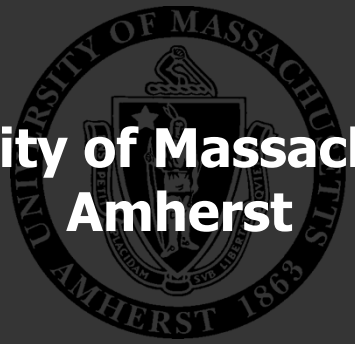
TFS: A Transparent File System for Contributory Storage

James Cipar, Mark Corner, Emery Berger

http://prisms.cs.umass.edu/tcsm



University of Massachusetts, Amherst

Contributory Applications

Users contribute resources from local machine for other's use

Contributory Storage

In desktop systems, many user's disks are half empty

- Typically 50% free disk space [Douceur 99, Huang 05]
- In modern computer systems, often more than 50 GB

Proposals for contributory storage describe diverse applications

- Backup and archival storage: OceanStore, PAST, Pastiche
- Serverless network file systems: CFS, Farsite, Pasta
- Anonymous publication: Freenet

Contributory systems **could** store massive amounts of data

- If every Folding@home user contributed 10 GB: 1.7 Petabytes

Current State of Contributory Storage

Freenet – only widely deployed contributory storage system

- BitTorrent uses storage, but only for files user wants
- Other systems use small amounts of storage for working data

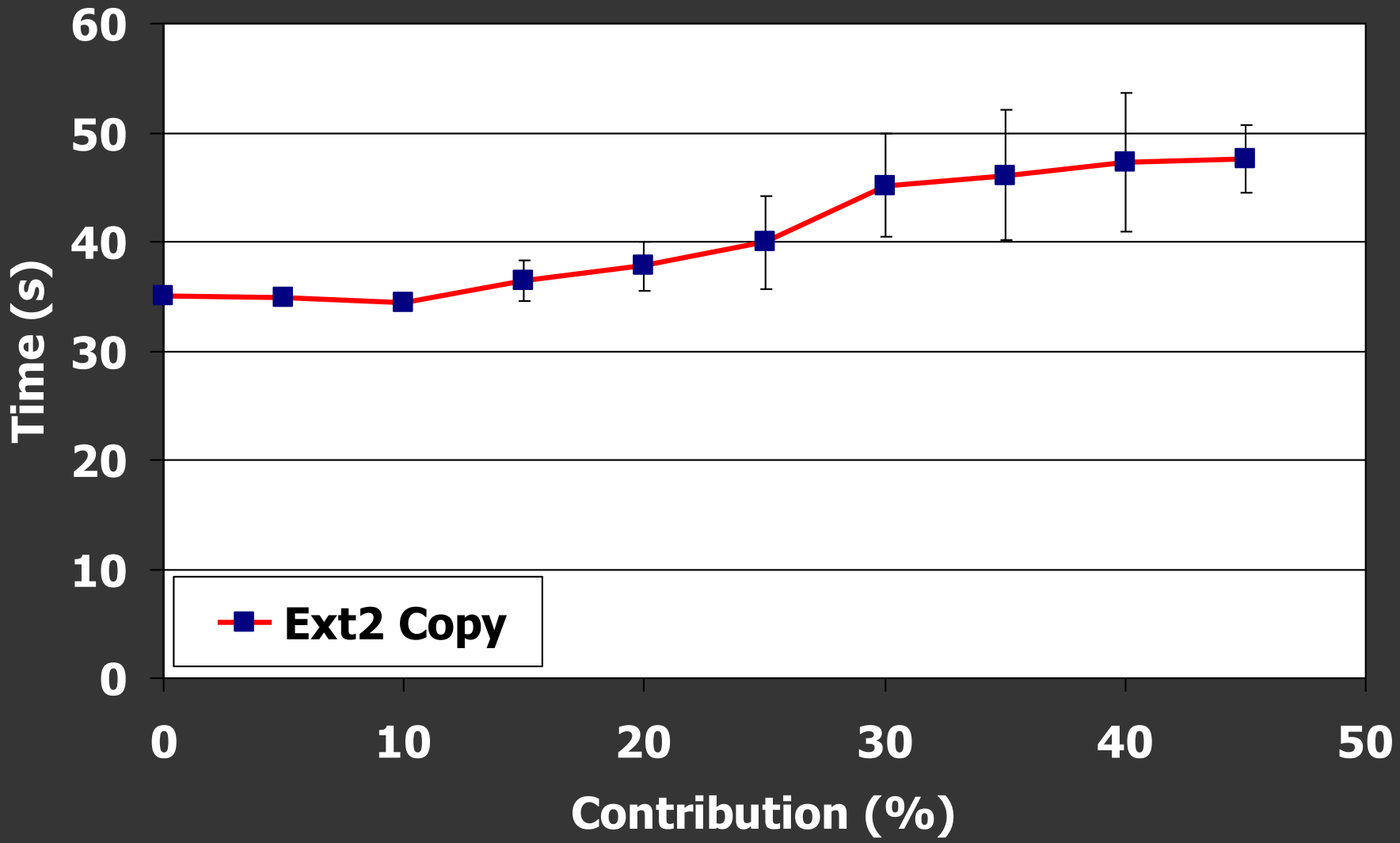
Biggest barriers: **loss of disk space** & **performance impact**

- 3 questions on Freenet FAQ ask about contributing less
- Contributed storage causes fragmentation in user's files
- File system aging can cause up to 77% degradation [Smith '97]

Result: contributory applications limited to very little disk space

- Ensures negligible impact on user, but…
- By default, Freenet uses 1 GB (out of ~ 50GB free = 2%)

Impact of Contributing Storage



As disk utilization increases, so does file fragmentation

The more you contribute, the worse performance gets

Our Solution: TFS

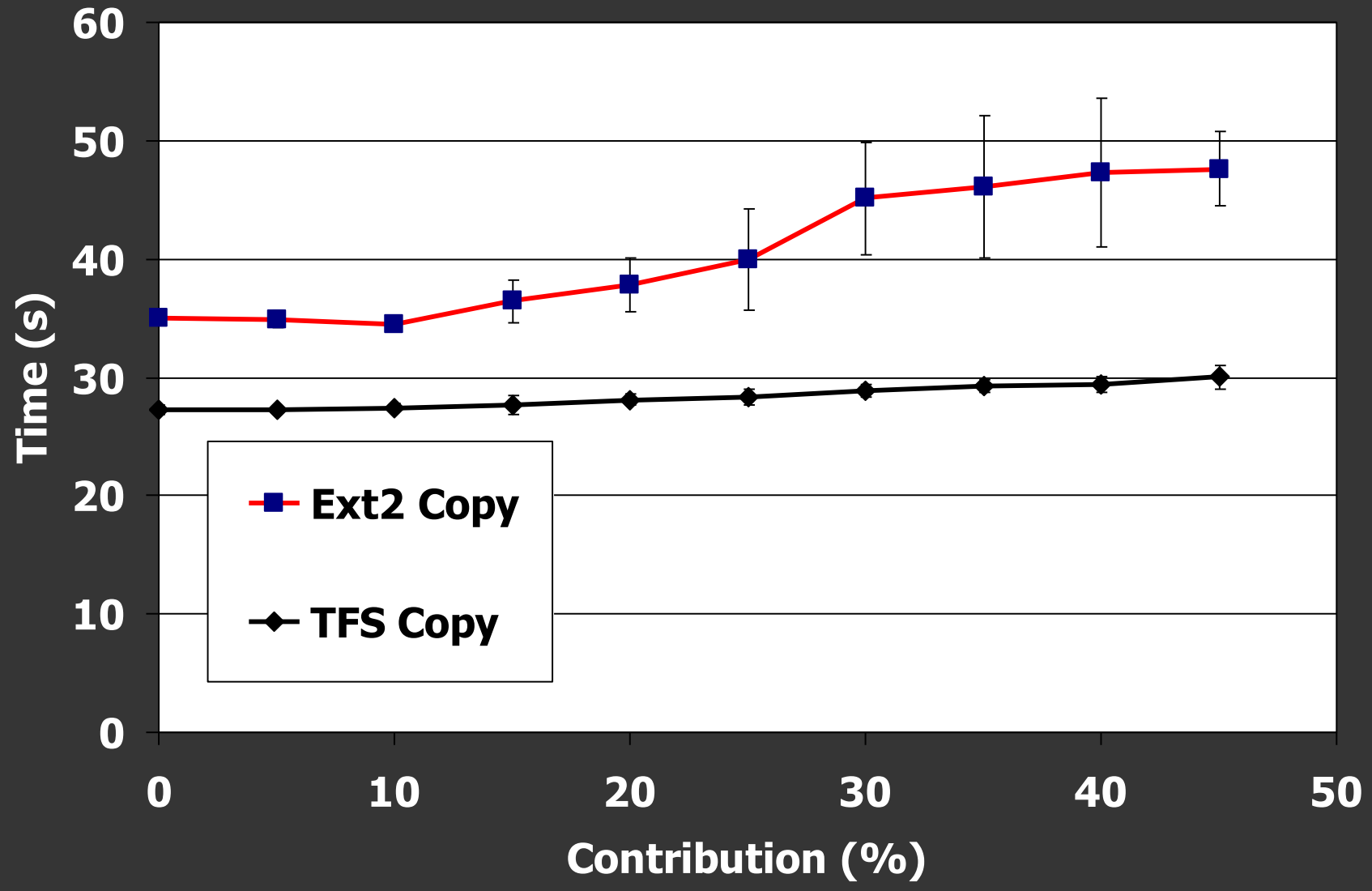
File system modification, provides 2 classes of files

- **Transparent files** do not interfere with ordinary files
- Files for normal applications behave the same
- On-disk file system to be used by distributed system

Contributory applications can use **all free disk space**

- Transparent files take **no noticeable space**
- **Negligible performance impact**

TFS and Contribution



TFS performance changes very little as space is contributed

Overview

Design principles

Disk allocation policies – preventing fragmentation

Performance concessions

Evaluation

- Local file system performance
- Effect on contributory storage applications

Design Principles

Contributory storage **virtually unnoticeable to user**

- No effect on allocation policy → no effect on performance
- **Transparent files cause no fragmentation**

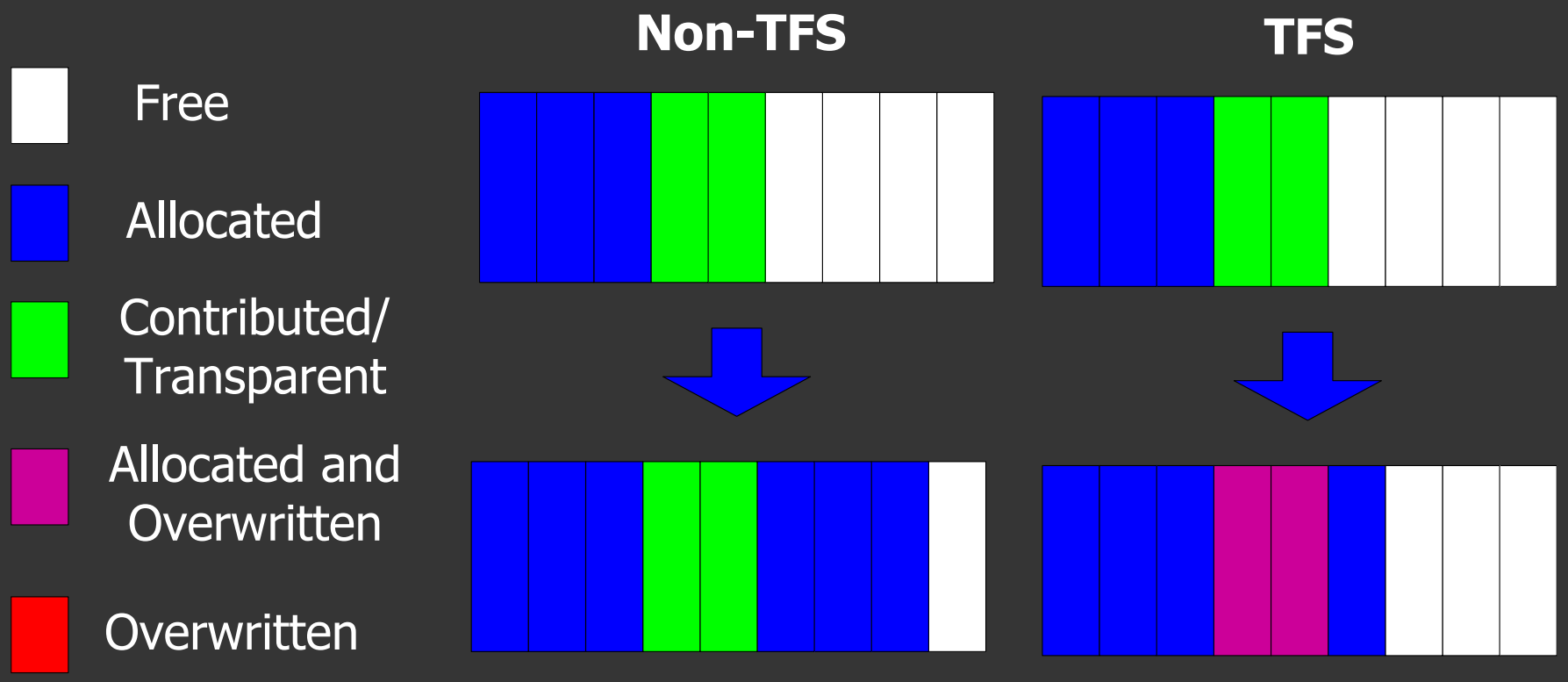
Contributory data may be overwritten to avoid interference

- Avoiding interference more important than data persistence
- Contributory applications should use replicas to prevent data loss

Minimal or **no modifications to contributory applications**

- Simple interface to mark directories containing transparent data
- Ordinary file semantics for transparent files

Avoiding File Fragmentation

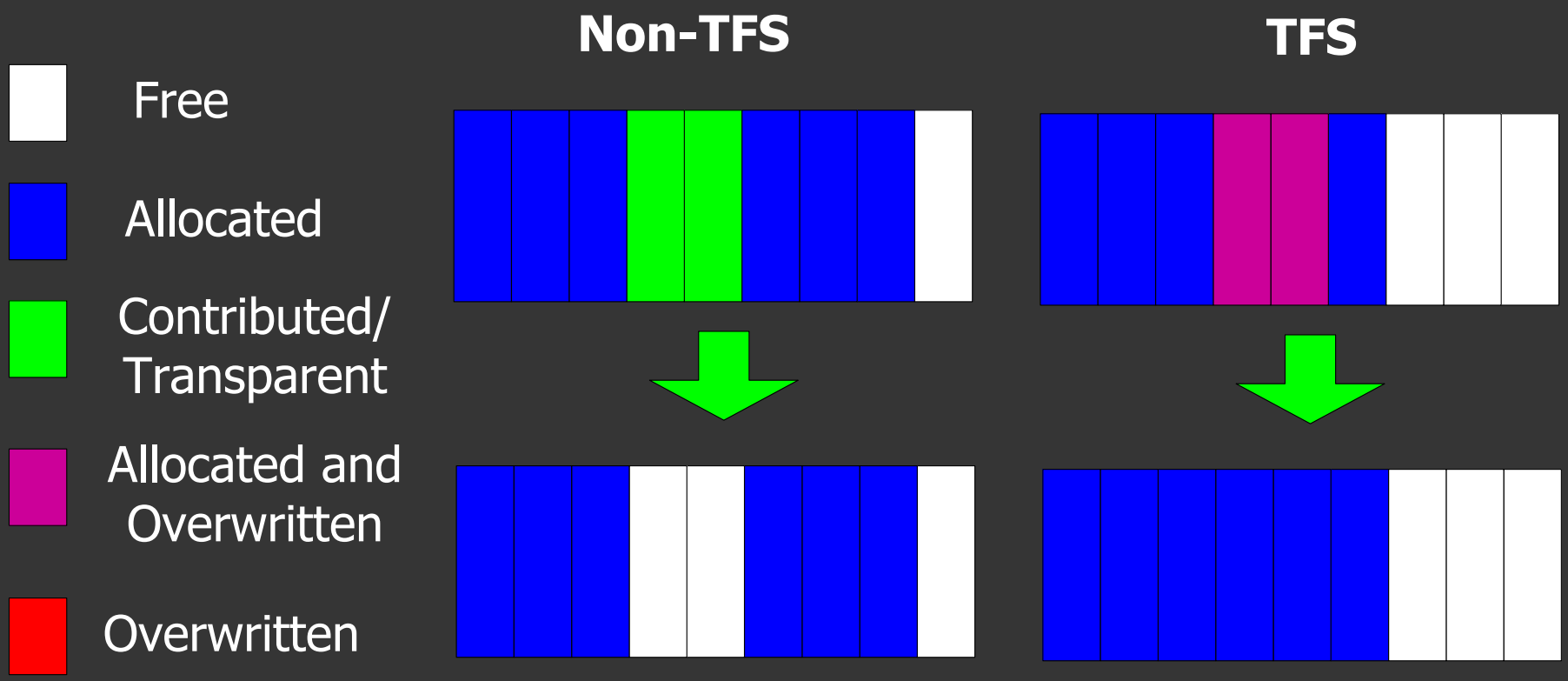


Blocks in TFS in one of five allocation states

TFS prevents contributed storage from causing fragmentation

Transparent file is now lost

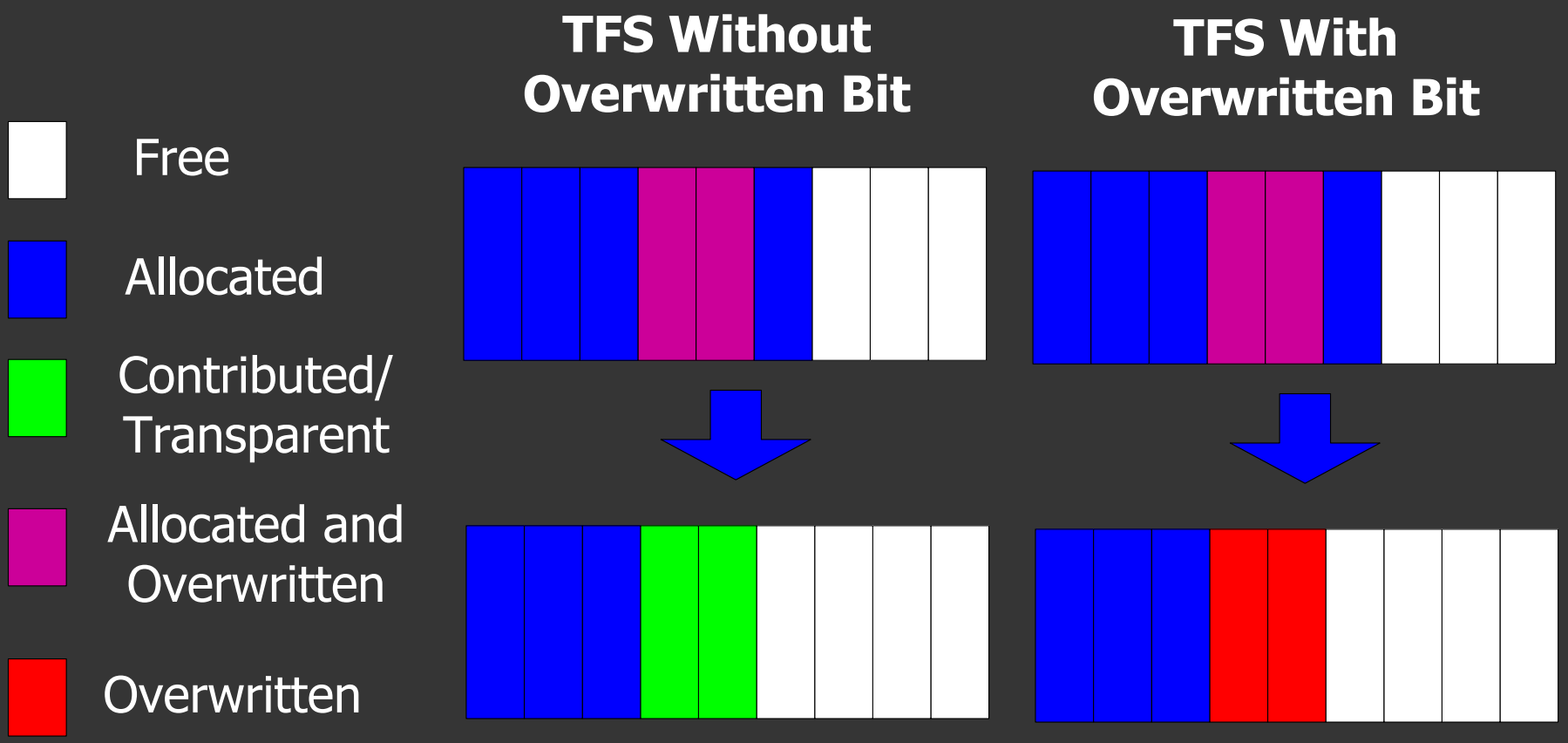
Avoiding Free Space Fragmentation



Freeing contributed storage does not leave gaps in free space

Contributing storage never adds to free space fragmentation

Why do we Need 5 States?



Ordinary file data overwrites contributed space, then is deleted

Overwritten state shows that data is no longer valid

Performance Concessions

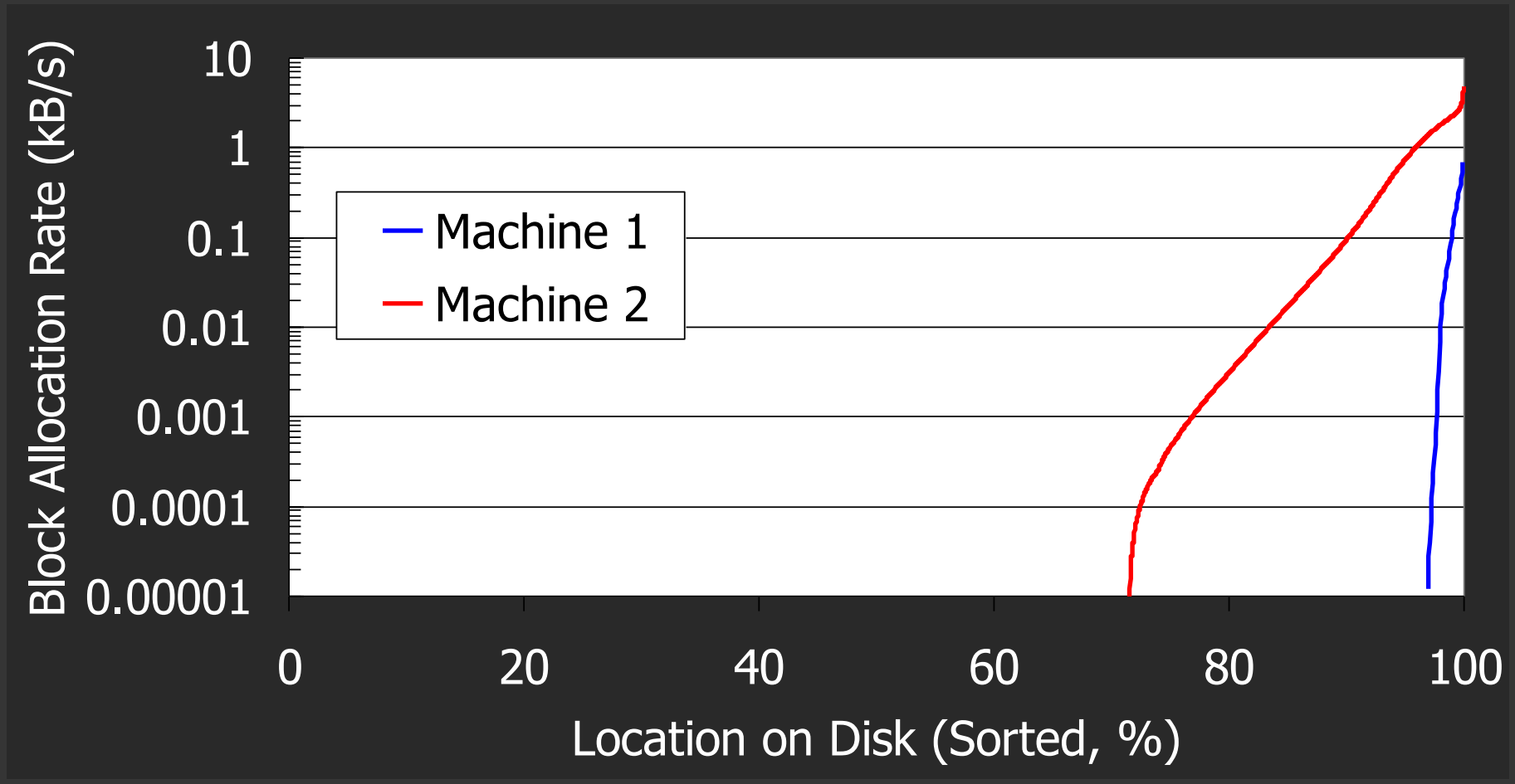
Open files cannot be overwritten

- Preserves normal file semantics for transparent files
- Assumes small amount of data being actively used at once

Transparent meta-data cannot be overwritten

- Directory entries, inodes, and indirect blocks are protected
  - Relatively small amount of storage
- Prevents large amounts of transparent storage being lost at once

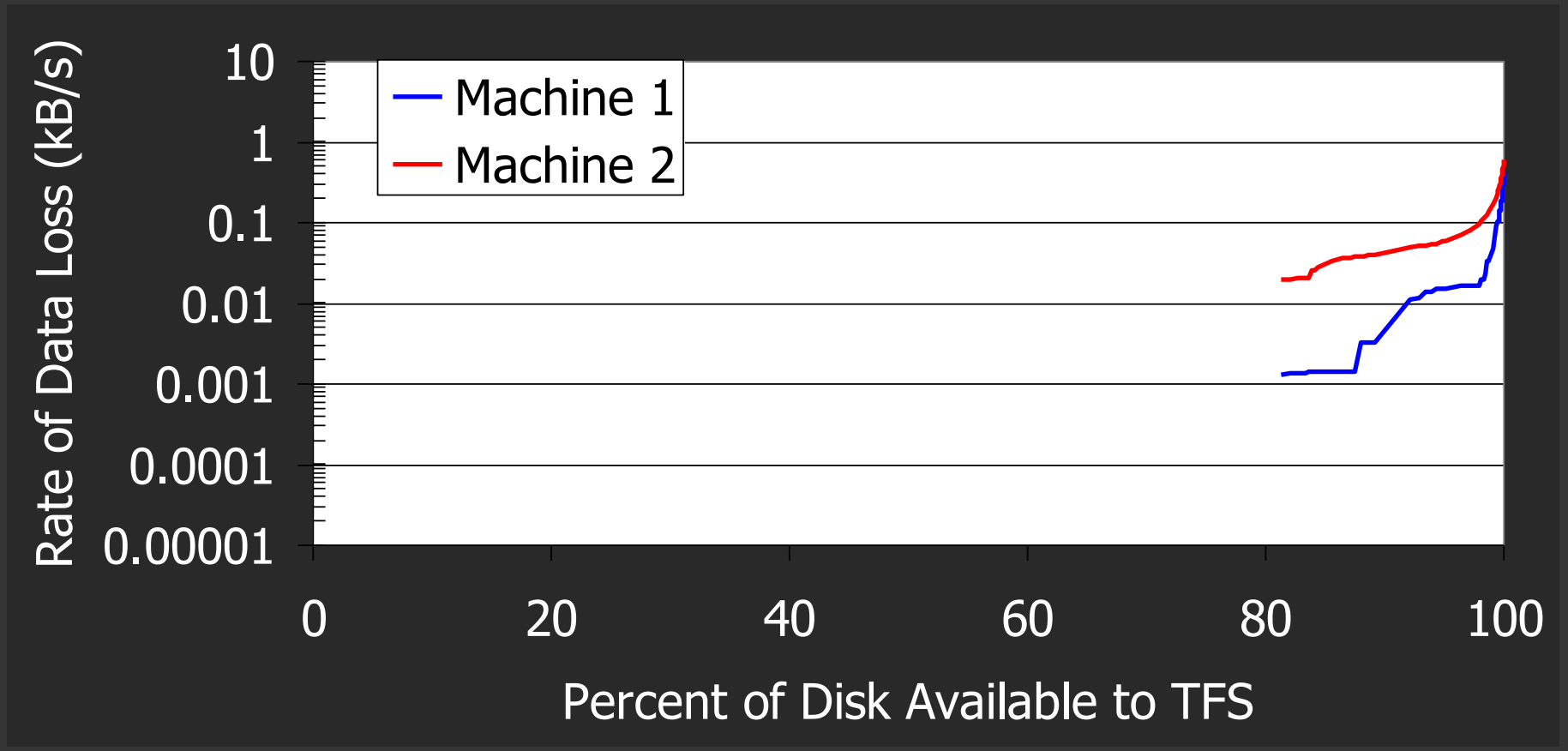
Allocation Locality



Block allocations exhibit high locality

Transparent data in “hot” areas likely to be overwritten often

Reducing Data Loss



TFS avoids placing data in hot parts of the disk

- Reduces rate of data loss

Traces allocation events to determine where to avoid

Overview

Design principles

Disk allocation policies – preventing fragmentation

Performance concessions

Evaluation

- Local file system performance
- Effect on contributory storage applications

Evaluation

Compare three methods of contribution

Small fixed-size contribution

- Represents current state of contributory storage
- Small enough to be unnoticeable to user

Dynamically managed watermarking

- Used by Elastic Quotas [Leonard '02], and FS² [Huang '05]
- Contributes more storage than fixed contribution, not all space
- Contributed storage automatically deleted when space is low

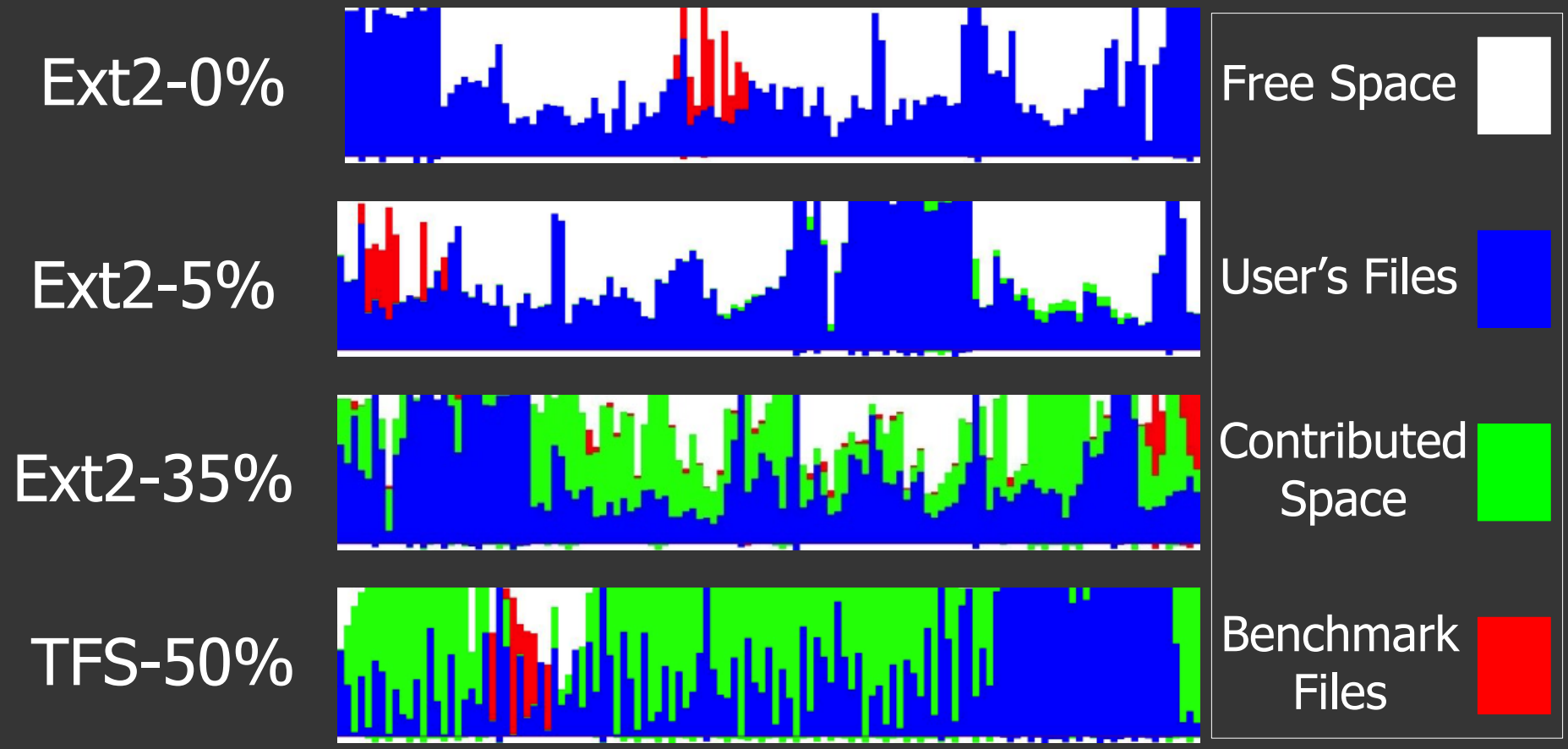
Contributed storage managed by TFS

- Can contribute all available storage

Experimental Setup

- Prototype of TFS based on Ext2
  - Linux kernel 2.6.13
- Tested on Dell Optiplex SX280
  - 512 MB Ram
  - 3 GHz Pentium 4
  - Disk type: Seagate ST-3160023AS
  - Disk Specs: 160 GB, 7200 rpm, Avg Seek 8.5ms,
    - 16383 cylinders, 16 heads, 63 sectors
  - Experiments performed on 10 GB file system
- Procedure:
  1. Disk half filled with simulated user data, taken from /usr
  2. Simulated contributory storage activity, file create and delete
  3. Copied benchmark data to file system
  4. Rebooted then ran Andrew Benchmark

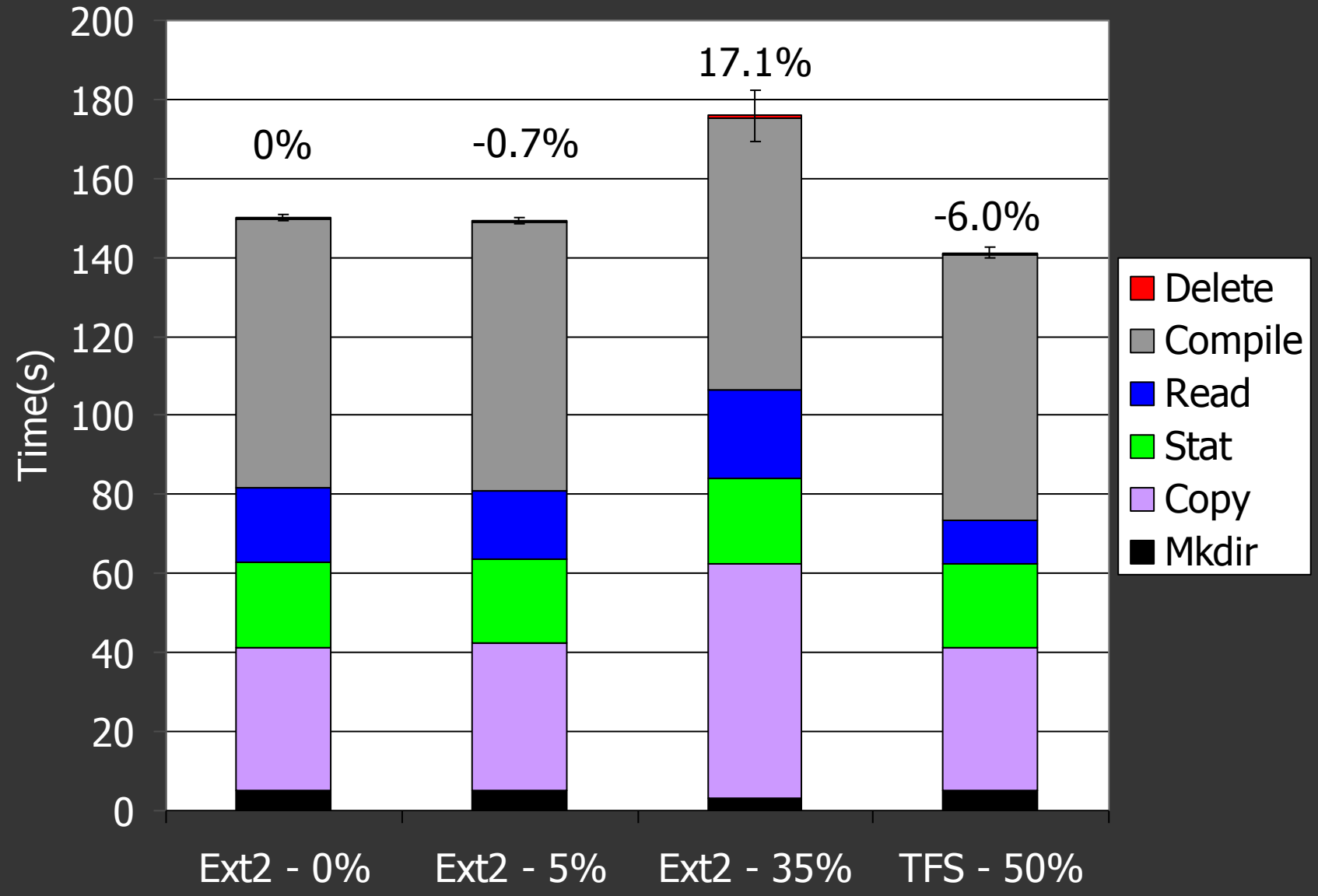
TFS Evaluation: Allocation Policy



Layout of benchmark data on disk while contributing storage

TFS prevents fragmentation while contributing all available space

TFS Evaluation: Performance



TFS preserves performance while contributing all available space

Storage Capacity and Bandwidth

Analyzed utility of TFS using model storage system [Blake '03]

- Provide persistent, available storage service
- Use replication to prevent data loss and ensure availability
- Assume infinite burst bandwidth, but limited average bandwidth

When host leaves network, its data must be replicated

- Replication bandwidth is function of storage per host and churn
- More churn → more bandwidth needed
- Less bandwidth → less usable storage

TFS and Bandwidth

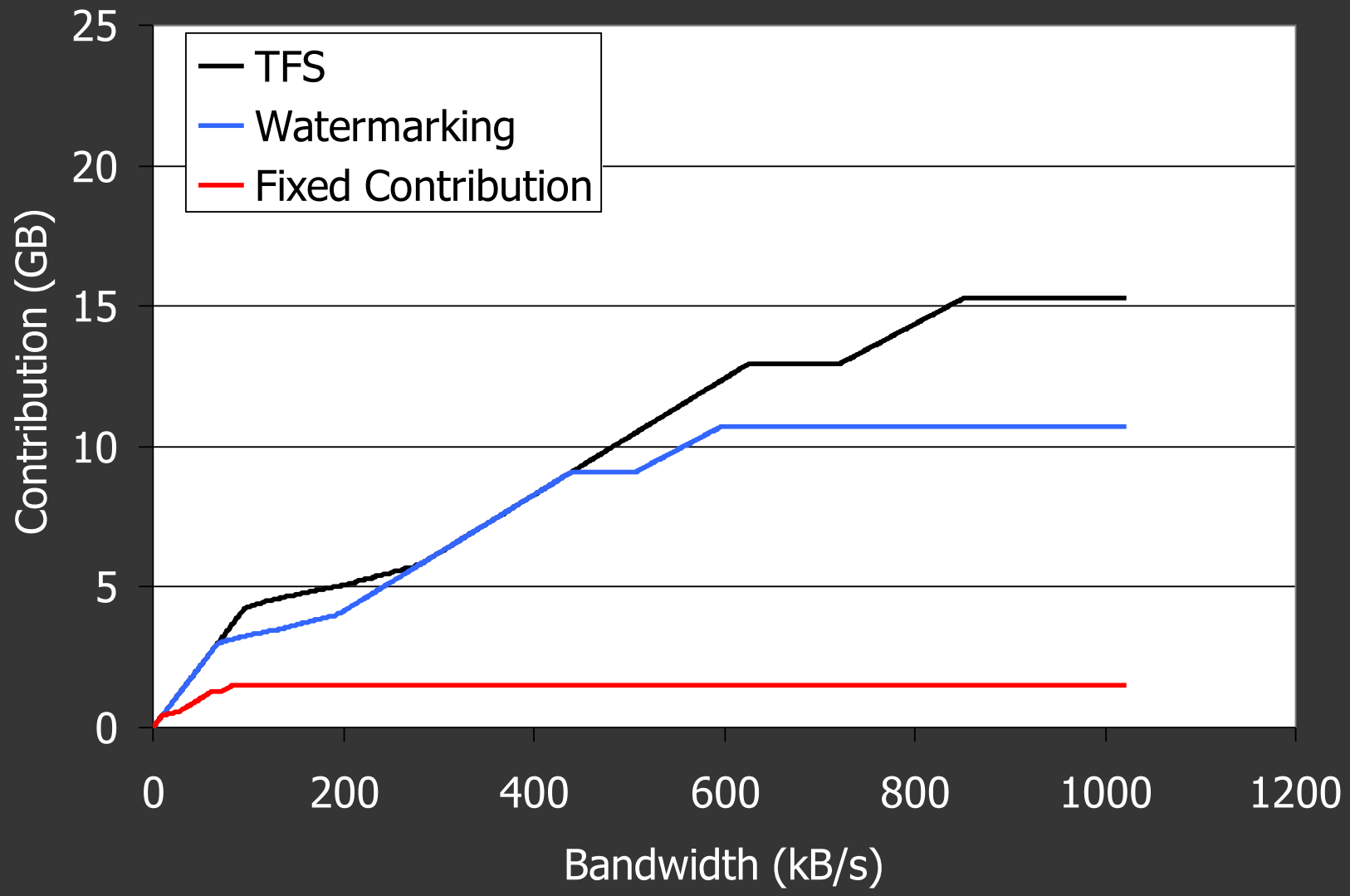
Trace-driven analysis of usable storage

- Used traces to find realistic churn levels [Bolosky ’00, Guha ’06]
- Used block allocation traces to determine rate of data loss in TFS

Given rate of data loss and amount of churn

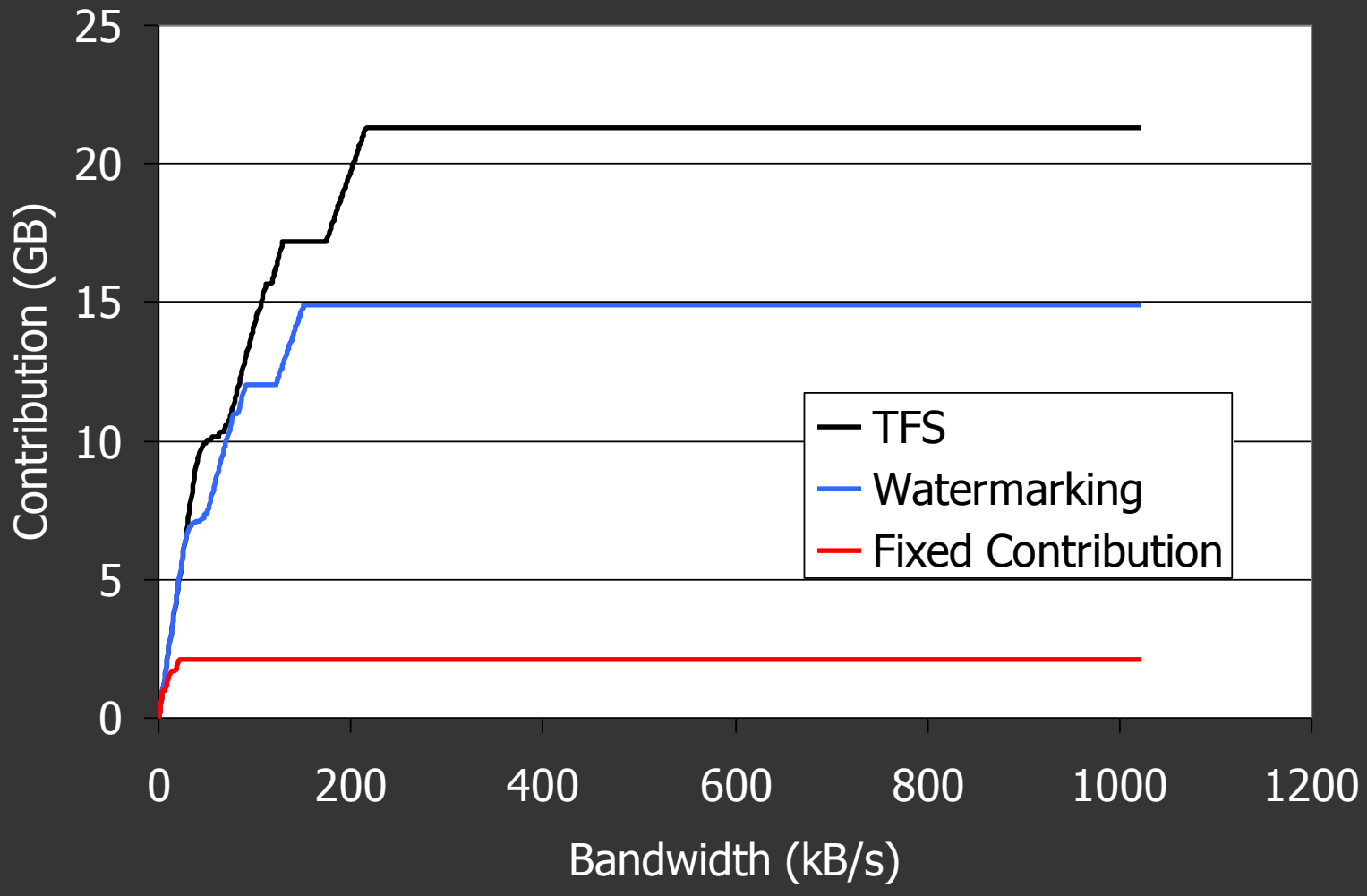
- Computed number of replicas needed for “five nines” of reliability
- Varied available bandwidth between 0 kB/s and 1200 kB/s
- Determined the maximum contribution per host
- BW mitigated through TCP-Nice, Diff-Serv, etc.

High Churn Network (Internet)



In a network with more churn, bandwidth limits storage

Low Churn Network (Corporate)



In a network with low churn, bandwidth does not limit storage

Conclusions

Contributory systems must not interfere with local performance

- Users will restrict or disable application
- Buffer cache can be managed by TMM [Cipar 06]

TFS allows contributory storage to use entire disk

- Very little interference
- No loss of storage for user

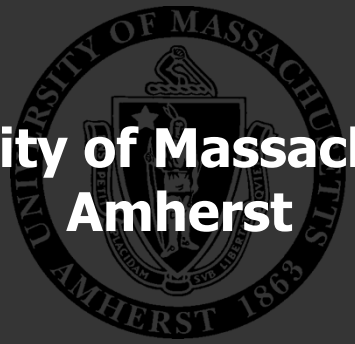
TFS + sufficient bandwidth → contributory systems can store much more data

http://prisms.cs.umass.edu/tcsm

TFS: A Transparent File System for Contributory Storage

James Cipar, Mark Corner, Emery Berger

http://prisms.cs.umass.edu/tcsm



University of Massachusetts, Amherst